STRONgrid Library: A Modular and Extensible Software Library for IEEE C37.118.2 Compliant Synchrophasor Data Mediation

Maxime Baudette[a], Luigi Vanfretti[b,*], Seyed Reza Firouzi[a]

[a]SmarTS Lab., KTH Royal Institute of Technology, Stockholm, Sweden
[b]Rensselaer Polytechnic Institute, Troy, NY

Abstract

The electric power grids contain highly dynamic behaviors that can be mitigated by exploiting Wide-Area Monitoring, Protection And Control (WAMPAC) technologies. These technologies rely on utilizing the synchrophasor measurements provided by the Phasor Measurement Units (PMUs). The IEEE C37.118.2 standard defines a method for real-time communication of synchrophasor data between PMUs, Phasor Data Concentrators (PDCs) and other applications.

This paper describes the Smart Transmission Grid Operation and Control (STRONgrid) library which serves as a real-time data mediator (i.e. interface) between the IEEE C37.118.2 protocol and applications consuming PMU measurements. The STRONgrid library packages all the necessary components to let the researchers focus on the development of synchrophasor applications in higher level programming environments (e.g. LabVIEW).

Keywords: IEEE C37.118.2, PMU, PDC, Synchrophasor, WAMPAC

*Corresponding author
Email address: luigi.vanfretti@gmail.com (Luigi Vanfretti)

1. Motivation and significance

Electric power systems throughout the world have seen highly dynamic behaviors with the connection of always more intermittent generation units such as renewable energy sources. The rapid development of smart grid technologies based on Information and Communication Technologies (ICTs) is an opportunity to address such new challenges [1].

Time-synchronized phasor (i.e. synchrophasor) measurements are sub second level information of the voltage and current conditions on the grid. These measurements are estimated and reported by the high-precision sensors of Phasor Measurement Units (PMUs) that are equipped to use a common and precise clock. PMU measurements are commonly used in Wide-Area Monitoring, Protection And Control (WAMPAC) systems as one of the building blocks for smart grids [2].

The potential of WAMPAC systems can be unlocked by designing, implementing, and testing innovative synchrophasor-based software and hardware applications. In order to acquire the real-time synchrophasor data, such applications communicate with PMUs or Phasor Data Concentrators (PDCs). The methods for exchange of synchronized phasor measurement data, defining the messaging specifications for real-time communication between PMU, PDC and other applications are standardized as IEEE C37.118.2 [3] or IEC 61850-90-5.

In the IEEE C37.118.2 protocol, synchrophasor data transfer is handled by means of exchanging four message types of: 1) Data, 2) Configuration, 3) Header and 4) Command Frames. The full details of the protocol can be found in the standard [3], or in a more concise interpretation in [4, 5]. A typical IEEE C37.118.2 message exchange mechanism, which has also been implemented in STRONgrid library, is presented in Fig. 1.

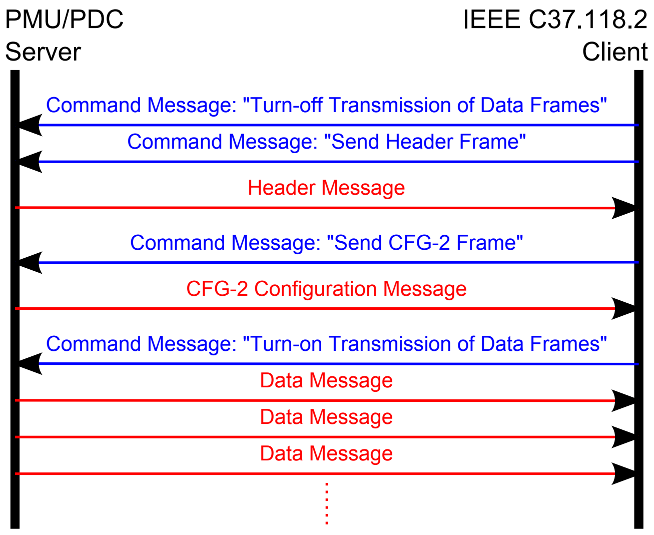

PMU/PDC Server
IEEE C37.118.2 Client
Command Message: "Turn-off Transmission of Data Frames"
Command Message: "Send Header Frame"
Header Message
Command Message: "Send CFG-2 Frame"
CFG-2 Configuration Message
Command Message: "Turn-on Transmission of Data Frames"
Data Message
Data Message
Data Message

Figure 1: IEEE C37.118.2 Message Exchange Mechanism

In this mechanism, the C37.118.2 client initially sends a *Command* message to stop transmission of *Data* frames by the PMU/PDC server. Then by sending another *Command* message, the client requests the server for the descriptive information available in the *Header* message. Before starting to receive the *Data* messages containing real-time synchrophasor data, the client needs to receive the PMU/PDC configurations available in the *Configuration Type-2 (CFG-2)* message. Once the *CFG-2* message received and parsed, the PMU/PDC server initiates the transmission *Data* messages.

At the Analysis Laboratory for Synchrophasor and Electrical energy Technology Lab. (ALSETLab), and its predecessor the Smart Transmission Systems Laboratory (SmarTS Lab) [6], the approach was to dissociate the PMU data mediation from the PMU data based applications. This is achieved by building a real-time data mediator as an interface to a higher level programing language more accessible to researchers in the field of power systems. Hence, researchers can focus on the development of synchrophasor monitoring and control applications on a personal computer (PC) (e.g. LabVIEW) or even embedded systems (e.g. National Instrument Compact RIO controller).

Several implementations have been developed, such as Bablefish [7] and Khorjin [8]. This paper focuses on the Smart Grid Synchrophasor Software Development Kit (S3DK) [9] that was developed as part of the Smart Transmission Grid Operation and Control (STRONgrid) project [10] The S3DK is comprised by two parts:

1. Real-Time Data Mediator (RTDM) core
2. LabVIEW user interface

The RTDM core is a Dynamic Link Library (called STRONgrid DLL) created from a set of sophisticated C++ source codes that implement the IEEE C37.118.2 standard. Thus, the STRONgrid Library will handle the communication with PMUs/PDCs through the Transmission Control Protocol and the Internet Protocol (TCP/IP) on the Microsoft Windows OS platform. The library also features an Application Programming Interface (API) that can be accessed by the user.

The LabVIEW user interface (called S3DK also), is a set of LabVIEW blocks that both provide a user interface to configure and initiate the PMU connection, as well as a set of functions to programmatically access the same functionalities and access the PMU data for further processing. The S3DK was developed to use the STRONgrid library through its API. More details on that LabVIEW user interface will be published in a future publication.

2. Software description

The STRONgrid library was developed as a real-time data mediator for further processing of PMU measurements in custom WAMPAC applications. Thus, it was designed as an intermediary program that translates PMU measurements sent using the IEEE C37.118.2 protocol, and delivers the message through an API. The connection to the PMU/PDC server is established by providing the library with the following information, as for any regular PMU application:

1. IP Address,
2. Port Number,
3. IDCODE (or PMU ID).

The interactions with library are carried out through the library's API, by sending successive commands. In the STRONgrid project, the goal was to develop a solution, the S3DK, dedicated to deliver PMU measurements in the LabVIEW development environment. In this context, the implementation of API was tailored to communicating with LabVIEW.

Although the current implementation of the API may not be fully practical to use with a different development environment for the PMU applications, there is nothing preventing further extensions of the API to support other programming languages. The library is, however, limited to the Microsoft Windows platform, as it relies on specific libraries for TCP/IP.

The STRONgrid DLL and its interactions with other software components can be summarized as illustrated in Fig. 2.

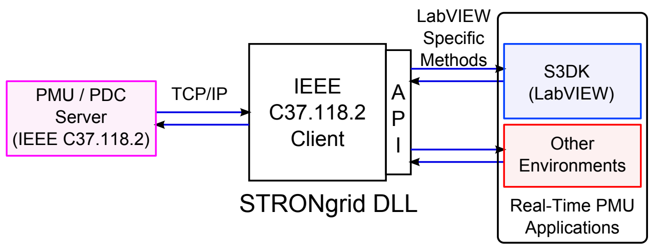


Figure 2: Strongrid IEEE C37.118.2 DLL Interface overview

2.1. Software Architecture

The STRONgrid library source code is delivered as a Microsoft Visual Studio project developed in C++. The code is structured in a modular architecture with four main blocks separating the main functionalities of the library:

- `StrongridBase`,
- `StrongridClientBase`,
- `StrongridDLL`,
- `StrongridDLLStressTest`.

The `StrongridBase` block contains the code and methods dedicated to interacting with and parsing the IEEE C37.118.2 protocol. It also implements the methods to encode messages according to the protocol's specifications.

The `StrongridClientBase` block implements the IEEE C37.118.2 client functionality as an object of the `PdcClient` class. It is in this object that the private methods emulating a PDC client are implemented with the necessary logic to establish a connection to a remote PMU/PDC, and stream the real-time PMU data. These methods also extract the information to make it available through the API.

The `StrongridDLL` block implements the API (i.e. public methods) that will be used by the client application to access the PMU/PDC data. Note that the currently implemented methods have been tailored to communicating with LabVIEW.

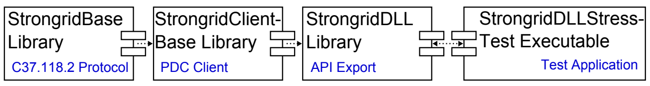


Figure 3: STRONgrid library architecture

Finally, the `StrongridDLLStressTest` block, implements a simple client application to test the DLL compiled from the other files. More details are presented in Section 3. The structure of the library is summarized in Fig. 3.

In the current version, all the functionalities to build a client application have been implemented, but the communication with the PMU/PDC server is limited to the TCP/IP protocol, and the library can handle the IEEE C37.118.2 protocol in its 2005 and 2011 versions. The implementation can, however, easily be extended to support additional protocols, such as the UDP/IP protocol for the remote connection, or the extension of the IEEE C37.118.2 protocol to a new revision. The code is also prepared to add a PDC server functionality.

2.2. Software Functionalities

The library includes all the necessary functionalities to enable a client application to stream PMU measurements in real-time. To achieve this, the the functionalities of the STRONgrid library can be summarized as follows:

- PMU/PDC connection configuration handling,
- Data from IEEE C37.118.2 protocol over TCP/IP parsing,
- Real-Time PMU data publishing (through API).

The detailed functionalities of the library are documented in the API, where each method corresponds to an individual functionality. There are two kinds of methods, the methods that can be polled to retrieve all available information (e.g. list the available PMUs), and the methods that can be provided with a set of parameters to retrieve a specific information (e.g. measurements from a single PMU). A non-exhaustive list of the methods available in the API is presented in Listing. 1.

Listing 1: Source code snippets showing some of the Strongrid DLL API methods

```
int connectPdc( char *ipAddress, int port, int32_t pdcId, int32_t*
    pseudoPdcId);
int disconnectPdc( int32_t pseudoPdcId);
int readHeaderData( int32_t timeoutMs, int32_t pseudoPdcId);
int readConfiguration( int32_t timeoutMs, int32_t pseudoPdcId);
int startDataStream( int32_t pseudoPdcId);
int stopDataStream( int32_t pseudoPdcId);
int readNextFrame( int32_t timeoutMs, int32_t pseudoPdcId);
int getPdcConfig( pdcConfiguration* pdcCfg, int32_t pseudoPdcId);
int getPmuConfiguration( pmuConfig* pmuconf, int32_t pseudoPdcId, int32_t
    pmuIndex);
int getPdcRealData( pdcDataFrame* rd, int32_t pseudoPdcId);
int getPmuRealData( pmuDataFrame* rd, PmuStatus* status, int32_t pseudoPdcId,
     int32_t pmuIndex);
int getHeaderMsg( char* msg, int maxMsgLength, int32_t pseudoPdcId);
```

In practice, the client application using the STRONgrid library should implement a workflow involving several of the available methods in the API. It is through this workflow that the functionalities as summarized above will be implemented.

As an example, a typical flowchart for a single thread - single PMU/PDC implementation is presented in Fig. 4.

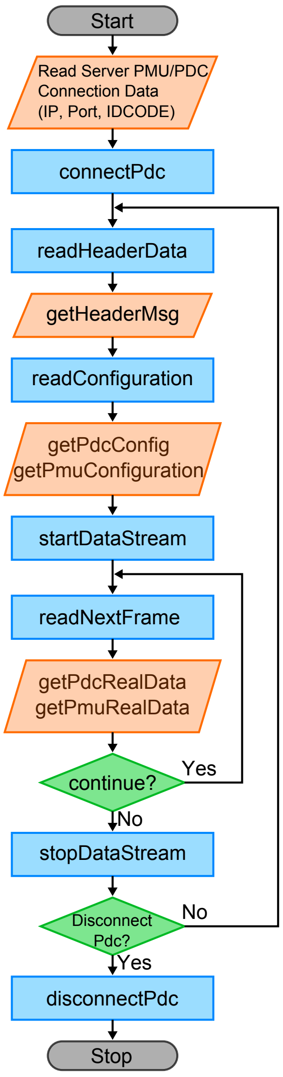


Figure 4: Strongrid library functionality flowchart

The workflow is initiated by providing the connection information when calling the `connectPdc` method. This will create a `PdcClient` object and establish a socket connection using the credentials passed as arguments. On success, the API returns `1` and a `pseudoPdcId`, uniquely identifying the PDC.

The `PdcClient` will be identified through the `pseudoPdcId`, when calling the following methods in the process. The data streaming requires to first read the *Header* frame, that is requested through the `readHeaderData` method. The DLL will then transmit a "Send Header Frame" *Command* message to request that information. The application can then access the information by calling the `getHeaderMsg` method. Similarly, the PMU/PDC configurations are requested by calling the `readConfiguration` method and accessed by using `getPdcConfig` and `getPmuConfiguration` methods.

At this point the PMU/PDC is fully identified and the connection established, the data streaming can be initiated by calling the `startDataStream` method. A loop can then proceed to consume the real-time data, calling the `readNextFrame` method, to decode the next IEEE C37.118.2 *Data* message, and the synchrophasor data are accessed using the `getPdcRealData` and `getPmuRealData` methods.

At any moment, the application can interrupt the data streaming with the `stopDataStream` method, which will send a "Turn-off Transmission of Data Frames" *Command* message to the remote PMU/PDC. At the end, the remote connection will be closed by calling the `disconnectPdc` method.

Note that the presented workflow includes only one remote PMU/PDC. More complex workflows can be built, and in the current implementation, the polling of the different PMU/PDC for new data is synchronous, through a synchronization layer. This allows the developer to focus on other aspects of the application.

3. Illustrative Examples

As previously described, the STRONgrid library seeks to facilitate the utilization of real-time PMU data in client applications. Thus, the examples presented here are client applications using the library.

3.1. StrongridDLLStressTest Executable

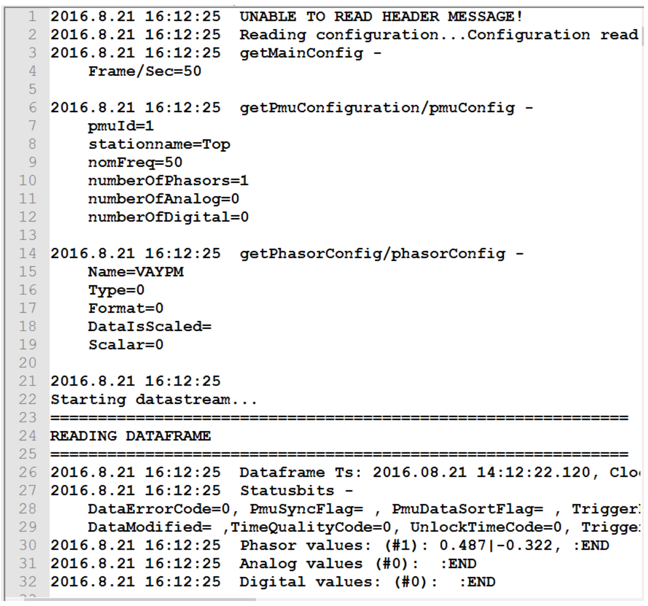
The `StrongridDLLStressTest` is a simple program developed as part of the STRONgrid library as a testing suite. Upon compilation of the STRONgrid library Visual Studio Project, both the DLL and the `StrongridDLLStressTest` executable will be created. This program will test the API methods of the DLL and attempt to establish a connection to a remote PMU/PDC to stream synchrophasor data. The program actually implements a workflow very similar to that of Fig. 4. All the test results are consigned in a log file to help diagnose any problem.

The executable was run once to produce the log file presented in Fig. 5. In this test, the PDC was streaming out the data from a single PMU, named *Top*. The *Top* PMU contained a single phasor named "VAYPM". The data streaming is also initiated during the testing process and the received data consigned in the log file.

Figure 5 shows that the IEEE C37.118.2 client application requested the Header information, but *Header* message was not provided by the PDC server (line #1). The PMU and Phasor *Configuration* data can be found in lines #6-19, and the *Data* message with the first Phasor value is printed in line #30.

3.2. S3DK LabVIEW Interface

The S3DK LabVIEW Interface is a set of LabVIEW VIs (i.e. library) that integrate the STRONgrid DLL with the LabVIEW development environment. The library seeks to alleviate the integration of PMU measurements by implementing the necessary data managements and operations. It enables the user to

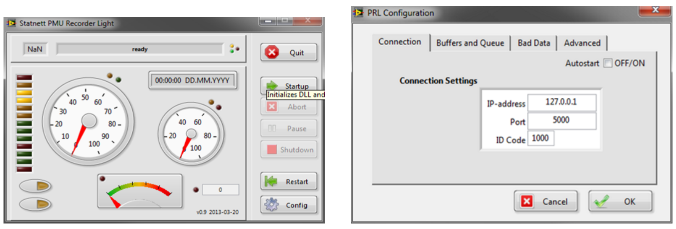


```
2016.8.21 16:12:25  UNABLE TO READ HEADER MESSAGE!
2016.8.21 16:12:25  Reading configuration...Configuration read
2016.8.21 16:12:25  getMainConfig -
     Frame/Sec=50

2016.8.21 16:12:25  getPmuConfiguration/pmuConfig -
     pmuId=1
     stationname=Top
     nomFreq=50
     numberOfPhasors=1
     numberOfAnalog=0
     numberOfDigital=0

2016.8.21 16:12:25  getPhasorConfig/phasorConfig -
     Name=VAYPM
     Type=0
     Format=0
     DataIsScaled=
     Scalar=0

2016.8.21 16:12:25
Starting datastream...
=============================================================
READING DATAFRAME
=============================================================
2016.8.21 16:12:25  Dataframe Ts: 2016.08.21 14:12:22.120, Clo
2016.8.21 16:12:25  Statusbits -
     DataErrorCode=0, PmuSyncFlag= , PmuDataSortFlag= , Trigger
     DataModified= ,TimeQualityCode=0, UnlockTimeCode=0, Trigge
2016.8.21 16:12:25  Phasor values: (#1): 0.487|-0.322, :END
2016.8.21 16:12:25  Analog values (#0):  :END
2016.8.21 16:12:25  Digital values: (#0):  :END
```

Figure 5: PDC log file: Output of `StrongridDLLStressTest` Executable

interact with the DLL and access the real-time PMU data by providing simple blocks.

The main components of the library are a graphical user interface (GUI) and a set of functions that can be integrated into an existingprogram. The S3DK GUI allows the user to directly interact with the DLL, as shown in Fig. 6. There the user can configure the connection information, initiate the connection and start the data streaming. The set of functions can easily be integrated into more complex programs consuming PMU measurements to automate the connection process and access the real-time data provided as LabVIEW data-types (see Fig. 7).



IP-address 127.0.0.1
Port 5000
ID Code 1000

Figure 6: S3DK Graphical User Interface (GUI)

The end goal of such library is to build more advanced program such as Wide-Area Monitoring System (WAMS) application using real-time PMU data. Figure 8 shows such an application with a frequency display, a frequency spectrum analysis and a “oscillation power meter”, see [11] for more details.

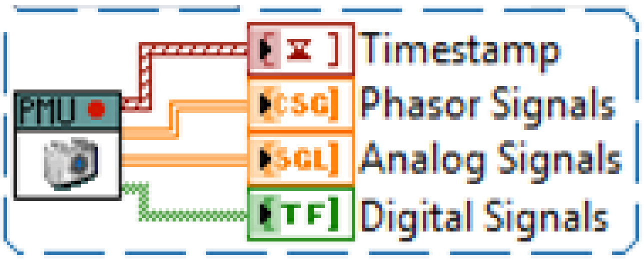


Figure 7: Delivery of the PMU data in LabVIEW Data types

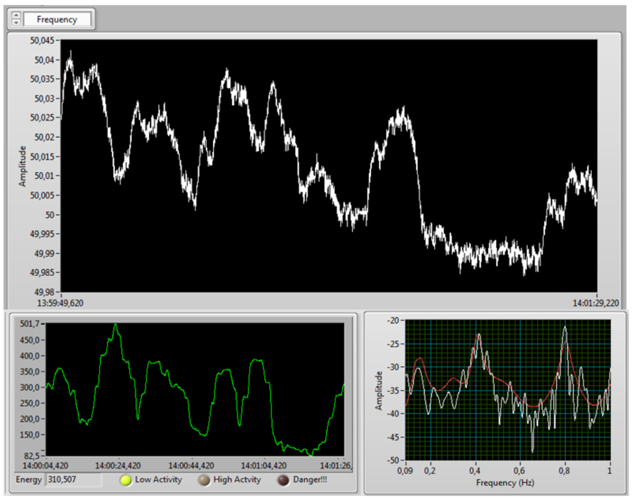


Figure 8: IEEE C37.118.2 Message Exchange Mechanism

4. Impact

As mentioned in Section 1, the research in smart grid applications will require to develop new and innovative WAMPAC systems. In the research, however, there is an obstacle to the development, and even more so prototyping/testing, of such solutions as the access to real-time PMU measurements is done through the IEEE C37.118.2 standard protocol, because researchers in the field don't necessarily know the technicalities behind this protocol.

The STRONgrid library seeks to address this issue by providing an open source real-time data mediator for the the IEEE C37.118.2 protocol. The implementation of this real-time data mediator was carried out in a low-level programming language (C++) to maximize its performances. The library is fully controllable through an API that was designed to be integrated in the LabVIEW development environment. Together with the S3DK LabVIEW Interface, it offers an attractive solution to power systems researchers by letting them design and prototype PMU-based applications in a high-level programming language.

In the literature, there are many ideas for developing new WAMPAC systems, but the examples of working prototypes or technology-demonstrators are still too few to have an impact in the actual adoption of these new technologies by the transmission system operators. The project seeks to facilitate the development of such prototypes.

5. Conclusions

The STRONgrid library was developed as a real-time data mediator to interface PMU/PDCs equipments using the IEEE C37.118.2 standard protocol and application consuming real-time PMU measurements. The library was developed together with a LabVIEW interface, which together build the Smart grid Synchrophasor SDK (S3DK). S3DK was successfully tested with the development of several prototype PMU applications.

The STRONgrid library implements already all the necessary functionalities to use it as a PDC client, receiving and decoding data from the IEEE C37.118.2 protocol. The included API is fully functional to work with LabVIEW together with the rest of the S3DK package.

The code can, however, be extended with additional methods in the API, to add the PDC server functionality, or even add the support for UDP data streaming.

Acknowledgements

The support of the following funding bodies is gratefully acknowledged:

- Statnett SF, the Norwegian Transmission System Operator.
- The EU funded FP7 project ***i**Tesla*
- The STRONg^2*rid* project by Nordic Energy Research.
- The STandUP for Energy collaboration initiative

The technical support provided by the following persons is sincerely acknowledged:

- Vemund Aarstrand, ENFO AS, Oslo, Norway
-

References

[1] J. Ekanayake, K. Liyanage, J. Wu, A. Yokoyama, N. Jenkins, Smart Grid: Technology and Applications, John Wiley & Sons, 2012.

[2] D. Mah, P. Hills, V. O. Li, R. Balme, Smart Grid Applications and Developments, Springer, 2014.

[3] IEEE standard for synchrophasor data transfer for power systems (2011). `doi:10.1109/IEEESTD.2011.6111222`.

[4] K. Martin, G. Brunello, M. Adamiak, G. Antonova, M. Begovic, G. Benmouyal, P. Bui, H. Falk, V. Gharpure, A. Goldstein, Y. Hu, C. Huntley, T. Kase, M. Kezunovic, A. Kulshrestha, Y. Lu, R. Midence, J. Murphy, M. Patel, F. Rahmatian, V. Skendzic, B. Vandiver, A. Zahid, An overview of the IEEE standard c37.118.2—synchrophasor data transfer for power systems, smart Grid, IEEE Transactions on (2014). `doi:10.1109/TSG.2014.2302016`.

[5] S. R. Firouzi, L. Vanfretti, A. Ruiz-Alvarez, H. Hooshyar, F. Mahmood, Interpreting and implementing iec 61850-90-5 routed-sampled value and routed-goose protocols for ieee c37.118.2 compliant wide-area synchrophasor data transfer, Electric Power Systems Research.

[6] L. Vanfretti, M. Chenine, M. Almas, R. Leelaruji, L. Angquist, L. Nordstrom, Smarts lab — a laboratory for developing applications for wampac systems, in: Power and Energy Society General Meeting, 2012 IEEE, 2012, pp. 1–8. doi:10.1109/PESGM.2012.6344839.

[7] L. Vanfretti, I. A. Khatib, M. S. Almas, Real-time data mediation for synchrophasor application development compliant with IEEE c37.118.2, in: 2015 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), Institute of Electrical and Electronics Engineers (IEEE), 2015. doi:10.1109/isgt.2015.7131910.

[8] S. R. Firouzi, L. Vanfretti, A. Ruiz-Alvarez, F. Mahmood, H. Hooshyar, I. Cairo, An iec 61850-90-5 gateway for ieee c37.118.2 synchrophasor data transfer, 2016.

[9] L. Vanfretti, V. H. Aarstrand, M. S. Almas, V. S. Peric, J. O. Gjerde, A software development toolkit for real-time synchrophasor applications, in: 2013 IEEE Grenoble Conference, Institute of Electrical and Electronics Engineers (IEEE), 2013. doi:10.1109/ptc.2013.6652191.

[10] M. S. Almas, M. Baudette, L. Vanfretti, S. Lovlund, J. Gjerde, Synchrophasor network, laboratory and software applications developed in thestrongrid project, in: 2014 IEEE PES General Meeting | Conference & Exposition, Institute of Electrical and Electronics Engineers (IEEE), 2014. doi:10.1109/pesgm.2014.6938835.

[11] L. Vanfretti, M. Baudette, J. L. Domnguez-Garca, A. White, M. S. Almas, J. O. Gjerde, A pmu-based fast real-time sub-synchronous oscillation detection application, in: 2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC), 2015, pp. 1892–1897. doi:10.1109/EEEIC.2015.7165461.

Current code version

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v1.0.0
C2	Permanent link to code/repository used for this code version	`https://github.com/ALSETLab/S3DK-STRONGgrid`
C3	Legal Code License	GPL 3.0
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	C++
C6	Compilation requirements, operating environments & dependencies	Visual Studio 2017, cmake
C7	If available Link to developer documentation/manual	
C8	Support email for questions	luigi.vanfretti@gmail.com

Table 1: Code metadata (mandatory)